R-Tree based Searching and Ranking on Spatial Data

Ritty Jacob
Graduate Student Member,ACM
Viswajyothi College of Engg.&Technology

Kala M Karun
Graduate Student Member,ACM
Viswajyothi College of Engg.&Technology

ABSTRACT

Spatial databases have enormous number of applications, especially in mobile and wireless communications. Range queries are one of the best available tools to retrieve useful information from these databases. Range query usually returns large results. These results are neither communication effective nor informative. Finding spatial data that fit best for the intended use is the main challenge of all spatial search engines. Result of search for spatial data is a list of all items indicated as relevant by the algorithm used in the search engine. Depending on the type of input to the system, various techniques can be used for ranking the results. In order to address these problems, we propose an idea of r-tree based searching and ranking. Such an r-tree based searching & ranking reduces the costs of communication, increases the Usefulness, and also provides interactive exploration. Proposed system defines that the handheld device query will be provided effectively and nontrivial algorithm found good results approximately.

General Terms

Range Queries, Ranking, Architecture, Rectangles, Point set

Keywords

Spatial databases, Searching, Algorithms.

1. INTRODUCTION

Now a day's spatial databases have a large number of applications especially in the field of mobile and wireless communications and embedded systems. These revolutions are mainly related with the changes in the internet. One of the main applications is, people can trace their location via mobile phones or PDAs.Handheld devices like sensors and satellites collect a huge amount of spatial data. But these devices have faced a number of problems related with the bandwidth and computing power. Range queries are the main tools to retrieve information from these spatial databases. However results should be light and usable to the end user. It is a challenging, but an interesting task. Similar challenges have been faced by large relational databases and OLAP [1] data warehouses while processing large queries. All of these processing requirements will finally be focused on one goal to provide light, usable representations of the query results, such that an interactive query process is possible. That is in each possible stage of a long-running query evaluation concise representation [2] of the final query results should be returned.

The approximate representation of join queries in a relational database can be considered as a random sample of the final query results. For this case random sampling is not a good solution, the goal is completely different. Here light refers to the fact that size of the query results should be as small as possible. Mobile computing and embedded systems have very limited client-server bandwidth. This limitation leads to the prevention of communication of query results with a large size. The situations are same for applications with PCs over the Internet. Among different choices in order to choose the service of a product response time is a critical factor. But long response time may be a burden to the users.

The computational and memory resources are often limited in client devices. This leads to the inefficiency in processing large query results. These are the situations in the case of computing and embedded systems. The large query results often create a large computational and I/O burden on the server. In order to speed up query processing various efficient indexing structures are adopted by the database indexing community, but the query processing overhead is more due to the result size. The processing cost on the server can be reduced by the small representation of the whole query results. The compression technique solves the problem of large query results, but it cannot address the problem of resource limitation. Usability refers to the amount of meaningful knowledge, which a user can derive from the query results.

Large query results often create an information overloading problem. It results more harm than good to users. As an example, suppose a user types a query in order to find the good restaurants in his/her locality. In order to choose a good alternative, the query results are not apt. The small screen of a handheld device may display the large results in an overlapping fashion. So it is very difficult to differentiate between these results. The usability can be actually improved by representing the results in some specific size. The usability feature leads light to another important quality known as interactiveness.It has more importance than the usability.

Interactiveness refers to the capability that the user can refine the query results and provide feedback to the server. This property has more importance in the case of processing large query results. This helps users to go deeper into specific regions of a large query region. In the above mentioned example, the user can further refine the query to find out areas with high concentration of restaurants, more specifically Chinese Vs Indian restaurants. The user can further refine their query results.

1.1 Problem Definition

It introduces the concept of document searching and ranking based on concise range query results, where concise collectively represents the light, usable, and interactive requirements. Here, a point set can be represented using a collection of bounding boxes and the associated count of these point sets as a concise representation entire set. The user can specify the size of query according to the bandwidth and/or computing power of the mobile device, and can retrieve the query results in a concise form satisfying his/her constraints. At, the same time user can also fix the information loss and enquire for a concise representation with a minimum size. The methods term frequency and inverse document frequency are used to search and rank concise range query as text and spatial relevance.

2. LIMITATIONS OF OTHER TECHNIQUES

The main objective is to range concise query from the user to maximize the efficiency of handheld systems & to improve the document searching and ranking efficiencies. In the existing system constrain range of queries were extracted from database using sql query processing. It is very time consuming and also not much efficient. The information loss is also high. Proposed system defines that the handheld device query will be provided effectively and nontrivial algorithm found good results approximately.

3. QUERY PROCESSING WITH R-TREES

During the answering of concise range query, spatial indexing techniques were adopted by the database server in order to evaluate the query. R-tree provides such an indexing mechanism while evaluating the query results. It mainly follows the idea of bounding boxes .Bounding boxes collectively contain all the points inside the query region. There is a count associated with each of these bounding boxes, indicates the number of points associated with that bounding box. Then group these identified bounding boxes into the specified size in order to obtain the concise range query results. While answering a query, the R-tree based indexing will significantly save the CPU and I/O cost. So the concise range query will override the traditional techniques in many aspects. It not only improves the efficiency but also solves the problems of usability and bandwidth.

R-tree is the most widely used spatial index structure [3]. The R-tree [4] has the following structure. Suppose B be the disk block size. Then first forms a minimum bounding rectangle (MBR) by grouping *B* points in the considering area; these grouped points will be stored at a leaf on the R-tree. Further grouping of MBRs has been applied at each level until there exists only one. Each node *u* in the R-tree is associated with the MBR enclosing all the points stored below, denoted by MBR (*u*). Each internal node stores the MBRs of all its children.

The R-tree can be used to process the standard range query as follows. Initially starts from the root of the R-tree, and then check the MBR of each of its children. Then finds the nodes whose MBR intersects or falls inside the query region by recursively visiting the node. Simply return all the points stored that are inside the query region when we reach a leaf.

In R-tree MBRs are a good representation of the query results and it always tries to group close objects together with these MBRs. MBRs do not constitute a good concise representation of the query result, since the primary goal of the R-tree is fast query processing.

4. PROPOSED SYSTEM ARCHITECTURE

Proposed system defines that the handheld device query will be provided effectively and nontrivial algorithm found good results approximately. This technique is proposed on real world data in particular range on spatial area. Efficient document searching and ranking are applied on the concise range query. Creates R-tree structure to range user query, clustering query results based on the user query with spatial relevance's. Filtered text data will be filled on space-curve using Hilbert curve space-filling [5] method and user constrained query resulted using BFS search algorithm.

The system creates an r-tree structure to range user query, clustering query results based on the user query with spatial relevance. It follows some nontrivial algorithms to find a good approximate result which minimizes the information loss. The system is very much efficient and it demonstrates on real world data.

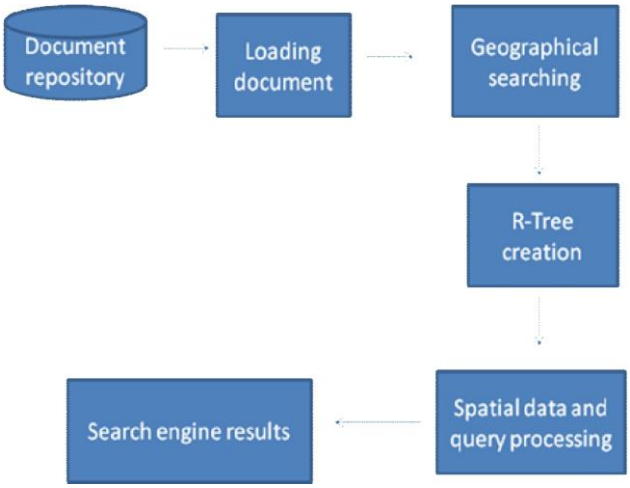

Figure 1: System architecture

Figure 1 shows the proposed system architecture. In this system we can create an R-tree based on the concise query range. Then the document searching and ranking methods are applied on this range query results. We use methods term frequency and inverse document frequency for search and rank query as text and spatial relevance. Term frequency means relevant words are retrieved from the database based on user query.

The system mainly includes the following functional units: loading documents, geographic searching, R-Tree creation, spatial data and query processing, search engine results.

A. Loading Documents

This unit is actually for loading the input data set. Loading documents with both spatial and textual data to be used for search engines. These data's includes the document set of some spatial areas. These data are stored in some spatial data

stores. Spatial database are the collection of geometry information such as polygons and lines. The user's need will be taken as input. For example, if the user wants the spatial specifications such as the hotels within 3 km. The collected spatial data are provided as an input**.**

B.Geographic Searching

Geographical searching includes textual and spatial relevance [6]. Textually and spatially relevant documents will be separated by query keywords. The system actually performs the geographic document searching and ranking based on textual and spatial data relevance. It uses the methods term frequency and inverse document frequency [7] [8] to search and rank query as text and spatial relevance.

Term frequency (tf) means relevant words are retrieved from the database based on user query. Inverse document frequency (idf (w, D)) measures the specific word w in a document set D.

idf (w, D)=log(|D|/|{d|d£D^tf(w,d)>0)

The context of geographic document search, the idf of a word w, denoted by idf(w, D,S).
Text relevance of document d to q is defined as :

$$ø_q(d) = \sum(tf(w,d) \cdot idf(w, D, S)).$$

Spatial relevance of a document d, denoted as ø(d), depends on the type of the spatial relationships defined between a document location L d and a spatial scope S.

Enclosed ø (d) is set to 1 if the corresponding location if fully enclosed by the query scope,

$$ø(d) = \begin{cases} 1, \text{ if } Ld \in S \\ 0, \text{ otherwise.} \end{cases}$$

Overlapping ø (d) is set to the fraction of the document location that is covered by the spatial scope,

ø(d)=Area(Ld n S)/Area(Ld).

Proximity ø(d) is represented by the inverse of the distance between the center of Ld and that of S,

$$ø(d) = \begin{cases} 1/dist(Ld, S), \text{ if } Ld \in S \\ 0, \text{ otherwise.} \end{cases}$$

C.R-Tree Creation

This unit creates the R-tree structure for user queries and results. The user specified queries are extracted from the R-tree structure. R-tree structure mainly focuses to range user query, clustering query results based on the user query with spatial relevance's.

D.Spatial Data and Query Processing

R-tree based query processing is adopted here. The query region from R-tree can be taken as an input. The spatial data and queries can be processed by using range queries.

E.Search Engine Results

An efficient indexing for geographic document search will be specified for spatial queries. The results of search engine can be produced as output. The query result size significantly reduced as required by the user. The reduced size saves communication bandwidth and also the client's memory and computational resources, which are of highest importance for mobile devices. Although the query size has been reduced, the usability of the query results has been actually improved. By using the R-tree-based algorithms a concise range query can be processed much more efficiently than evaluating the query exactly, especially in terms of I/O cost.

5. RESULTS

We have implemented the system using R-tree to index all the points in the data set. Then compare against *k*-means clustering algorithm, and MinSkew histogram [9]. We first obtain all the data points in the query region via R-tree, and then conduct either of the two methods to obtain clusters/partitions. The CPU time of MinSkew is the lowest (about 0.1 second), due to the low cost of constructing the histogram. However, it has the highest (worst) information loss among all approaches. The *k*-means algorithm incurs high CPU time (about 84 seconds) due to the data retrieval from the R-tree index and the clustering. Thus, *k*-means is not efficient in terms of CPU time (also true for the I/O cost), though the information loss is larger. The CPU and I/O costs are moderately low in the case of R-tree based indexing systems.

6. FUTURE WORK

The concise range query is quite a general concept. This general idea could naturally extend to deal with fuzziness in data [10], [11], [12], uncertainty and moving objects [13], [14], [15], etc. However, there exist many problems while designing efficient and effective query processing algorithms.

7. CONCLUSION

A new concept of ranking & searching of documents based on the concise range query results has been proposed in this paper, which addresses the following problems of traditional search engines. First, it reduces the query result size significantly as required by the user. The reduced size saves communication bandwidth and also the client's memory and computational resources, which are of the highest importance for mobile devices. Secondly, although the query size has been reduced, the usability of the query results has been actually improved. The concise representation of the results often gives the user more intuitive ideas and enables interactive exploration of the spatial database. Finally, we have designed R-tree based algorithms so that a concise range query can be processed much more efficiently than evaluating the query exactly, especially in terms of I/O cost.

8. REFERENCES

[1] Carlos Ordonez, Il-Yeol Song ,Carlos Garcia-Alvarado **,"**Relational versus non-relational database systems for data warehousing," DOLAP '10 Proceedings of the ACM 13th international workshop on Data warehousing and OLAP,2010.

[2] K. Yi, X. Lian, F. Li, and L. Chen, "The world in a nutshell: Concise range queries," IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 1, January 2011.

[3] Hanan Samet," Spatial Data Structures," Modern Database Systems: The Object Model, Interoperability, and Beyond, W. Kim, ed., Addison Wesley/ACM Press, Reading, MA, 361-385, 1995.

[4] A. Guttman, "R-trees: a dynamic index structure for spatial searching," in *SIGMOD*, 1984.

[5] B. Moon, H.v. Jagadish, C. Faloutsos, and J.H. Saltz, "Analysis of the Clustering Properties of the Hilbert Space-Filling Curve," IEEE Trans. Knowledge and Data Eng., vol. 13, no. 1, pp. 124-141, Jan.2001.

[6] Lee, K.C.K. , Baihua Zheng, Wang-Chien Lee , Dik Lun Lee, Xufa Wang," IR-Tree: An Efficient Index for Geographic Document Search," ," IEEE Trans. Knowledge and Data Eng,vol 23,pp 585-599,April 2011.

[7] Hung Chim , Xiaotie Deng ," Efficient Phrase-Based Document Similarity for Clustering," IEEE Trans. Knowledge and Data Eng,vol 20,pp 1217-1219,Sep 2008.

[8] Sulieman Bani-Ahmad, Ali Cakmak, and Gultekin Ozsoyoglu," Evaluating Publication Similarity Measures," Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2005.

[9] S. Acharya, V. Poosala, and S. Ramaswamy, "Selectivity Estimation in Spatial Databases," Proc. ACM SIGMOD, 1999.

[10] N. Dalvi and D. Suciu, "Efficient Query Evaluation on Probabilistic Databases," Proc. Int'l Conf. Very Large Data Bases (VLDB), 2004.

[11] R. Cheng, D. Kalashnikov, and S. Prabhakar, "Evaluating Probabilistic Queries over Imprecise Data," Proc. ACM SIGMOD, 2003.

[12] A.D. Sarma, O. Benjelloun, A. Halevy, and J. Widom, "Working Models for Uncertain Data," Proc. Int'l Conf. Data Eng. (ICDE), 2006.

[13] C.S. Jensen, D. Lin, B.C. Ooi, and R. Zhang, "Effective Density Queries on Continuously Moving Objects," Proc. Int'l Conf. Data Eng. (ICDE), 2006.

[14] P.K. Agarwal, L. Arge, and J. Erickson, "Indexing Moving Points,"Proc. Symp. Principles of Database Systems (PODS), 2000.

[15] Y. Tao, D. Papadias, and J. Sun, "The TPR*-Tree: An Optimized Spatio-Temporal Access Method for Predictive Queries," Proc.Int'l Conf. Very Large Data Bases (VLDB), 2003